Computer Science

[redacted]

Carnegie
Mellon

Distance Exponent: A New Concept for Selectivity Estimation in Metric Trees

Caetano Traina Jr. [1] Agma J. M. Traina[1] Christos Faloutsos[2]

March 1, 1999
CMU-CS-99-110

[1]Mathematics and Computer Science Institute
University of São Paulo at. S. Carlos
13560-970 São Carlos, São Paulo, Brazil

[2]School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

We discuss the problem of selectivity estimation for range queries in real metric datasets, which include spatial, or dimensional, datasets as a special case. The main contribution of this paper is that, surprisingly, many diverse datasets follow a "power law". This is the first analysis of distance distributions for metric datasets.

We named the exponent of our power law as the *"Distance Exponent"*. We show that it plays an important role for the analysis of real, metric datasets. Specifically, we show (a) how to use it to derive formulas for selectivity estimation of range queries and (b) how to measure it quickly from a metric index tree (like an M-tree).

We do experiments on many real datasets (road intersections of U.S. counties, vectors characteristics extracted from face matching systems, distance matrixes) and synthetic datasets (Sierpinsky triangle and 2-dimensional line). The experiments show that our selectivity estimation formulas are accurate, always being within one standard deviation from the measurements. Moreover, that our algorithm to estimate the *"distance exponent"* gives less than 20% error, while it saves orders of magnitude in computation time.

[1] On leave at Carnegie Mellon University (caetano@cs.cmu.edu) . His research was partially funded by FAPESP (Sao Paulo State Foundation for Research Support - Brazil, under Grants 98/05556-5).

[1] On leave at Carnegie Mellon University (agma@cs.cmu.edu). Her research was partially funded by FAPESP (Sao Paulo State Foundation for Research Support - Brazil, under Grants 98/0559-7).

[2] His research was partially funded by the National Science Foundation under Grants No. IRI-9625428 and DMS-9873442. Also, by the National Science Foundation, ARPA and NASA under NSF Cooperative Agreement No. IRI-9411299, and by DARPA/ITO through Order F463, issued by ESC/ENS under contract N66001-97-C-851. Additional funding was provided by donations from NEC and Intel.

Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Keywords: query selectivity estimation, metric spaces, range queries, index structures, metric structures

CMU-CS-99-110
Computer Science Department
School of Computer Science, Carnegie Mellon University

CMU-CS-99-110

Distance Exponent: A New Concept for Selectivity Estimation in Metric Trees

Caetano Traina, Jr.*, Agma J.M. Traina*, Christos Faloutsos

March 1999

Currently Unavailable Electronically

Keywords: Query selectivity estimation, metric spaces, range queries, index structure, metric structures

17 pages

We discuss the problem of selectivity estimation for range queries in real metric datasets, which include spatial, or dimensional, datasets as a special case. The main contribution of this paper is that, surprisingly, many diverse datasets follow a "power law". This is the first analysis of distance distributions for metric datasets.

We named the exponent of our power law as the "Distance Exponent". We show that it plays an important role for the analysis of real, metric datasets. Specifically, we show (a) how to use it to derive formulas for selectivity estimation of range queries and (b) how to measure it quickly from a metric index tree (like an M-tree).

We do experiments on many real datasets (road intersections of U.S. counties, vectors characteristics extracted from face matching systems, distance matrixes) and synthetic datasets (Sierpinsky triangle and 2-dimensional line). The experiments show that our selectivity estimation formulas are accurate, always being within one standard deviation from the measurements. Moreover, that our algorithm to estimate the "distance exponent" gives less than 20% error, while it saves orders of magnitude in computation time.

17 pages

*Currently on leave at Carnegie Mellon University from Mathematics and Computer Science Institute, University of Sao Paulo at S. Carlos, 13560-970 Sao Carlos, Sao Paulo, Brazil.

1 - Introduction

Multimedia (MM) systems are becoming increasingly useful in our day-to-day work, learning, leisure, etc., and increasingly important in computational environments. So, more and more database specialists are interested in supplying new methods and algorithms which can improve the capability to answer queries and which allow a better understanding of the embedding information. Multimedia applications typically use a large amount of complex data, such as sounds, images and videos. These data types are typically multidimensional or non-dimensional. Many other fields manipulate these kinds of data types, for example: Geographic information systems (GIS) working with images such as spatial description of cities, roads, or other spatial interest points on a 2D map; medical images where the datasets are obtained, for example, from CT or MRI scans, and are stated as point matrixes describing the level distribution of each slice obtained [UDUPA_91]; satellite images, also work with point matrixes describing the spacial distribution of the scanned objets [GONZALES_92].

The focus of this work is to estimate the selectivities in similarity queries in metric spaces. The typical query is: 'Find all the faces that are within 10 units of difference from a desirable face'. What we want to do is to estimate (a) the number of qualifying objects (faces, in the above example) and (b) the number of disk accesses required if this data is stored in an index tree. Of course, we are also interested in the selectivities and response times for all the other related types of queries (nearest neighbors, spatial joins, etc). Our major discovery is an empirical 'power law' that the accumulated distribution function of the distance seem to obey.

Notice that, selectivity estimation is important in query optimization in a RDBMS to answer multidimensional queries [FALOUTSOS_94], and to support data mining and data compression. In fact, there is a lot of work on the selectivity and disk access estimation for vector space, as we discuss in the survey section. The novelty of this paper is that it is the first to tackle *metric spaces* and metric index trees. Clearly, our work includes the vector spaces as a special case.

The paper is organized as follows. Section 2 provides some background for the subject, including the $\mathcal{M}$-tree structure. Section 3 states the fundamental observation and defines the concept of *distance exponent*, showing that it holds for a great variety of real data domains. In Section 4, we derive formulas to obtain the *distance exponent* of a given dataset and to estimate the selectivity for range queries. Section 5 presents the experimental results on real and synthetic datasets, illustrating the accuracy of our proposed formulas. Section 6 contains the conclusions of this paper.

2 - Survey

Index structures are fundamental tools to enable database systems to efficiently store and retrieve information from huge volumes of data. Images, sounds, and video data are typically multidimensional or even non-dimensional, corresponding usually to a large portion of the database. Thus, a suitable index structure is necessary for such type of data [GAEDE_98].

2.1 - Vector Spaces and Spatial Access Methods

A milestone for the multi-dimensional indexing or spatial access methods (SAM) is the R-Tree proposed by Guttman [GUTTMAN_84]. After that, many modifications and improvements of this structure were performed, bringing new solutions to the problem of index spacial data [SELLIS-87] [BECKMANN_90] [HELLERSTEIN_95]. The family of R-tree structures uses the data by itself to build the tree, i. e., the actual data is splited, using the geometric information to do it [PAPADIAS_95].

The analysis of index trees has been studied in the literature with regard to n-dimensional spaces. For instance, Faloutsos and Kamel [FALOUTSOS_94] [KAMEL_93] presents an analysis of R-trees for D-dimensional spaces, using the concept of correlation fractal dimension.

In multimedia applications it is very common to ask the database to return "all the pictures that are similar to this one". But, what is similar? A measurement to quantify similarity needs to be specified. The human brain intrinsically has the capability to understand the idea of similarity. However, in multimedia applications we need to resolve this problem computationally by storing the data for this kind of query. An index structure that organizes the data using the property of the dataset to be a metric space can bring us an answer to this question. Typically we extract a set of *n* features from an object, thus mapping it to a *n*-dimensional point. Then we can use a SAM to handle it .

The capability to index spatial data is necessary to answer range queries using this idea of similarity. Thus, if we could specify a distance method to measure such similarity, and if we could use such distance to build an index structure, we certainly could answer range queries, spatial joins [HUANG_97] and nearest neighbor queries [ROUSSOPOULOS_95]. However, occasionally we are given only a distance function, but no features. For such cases, only metric trees can be used. In the next section we will explain the idea of metric spaces as well as the spatial access method chosen (*M*-tree) to evaluate the number of qualifying points for range queries.

2.2 - Metric spaces and the metric tree

A *metric space* is a pair, $M=(D,d)$, where D is a domain of feature values and d is a metric distance as described follows. A distance function $d(x,y)$ for a metric space has to fulfill the following prerequisites: symmetry ($d(x,y) = d(y,x)$); non-negativity ($0 < d(x,y) < \infty, x \neq y$ and $d(x,x) = 0$) and triangle inequality ($d(x,y) \leq d(x,z) + d(z,y)$).

An index structure based on distances between objects in a metric space uses the above assumptions. Note that, in metric spaces different from Euclidian spaces, we do not use the information about the shape of the objects. The shape, texture, pattern, color of the objects and any other characteristics will be extracted, and afterwards the objects will be searched using the distance function stated for this kind of object. Thus, the distance function will be calculated according to the features extracted from the objects.

A metric space allows the storage of complex data using feature sets previously extracted from the dataset. In this paper we are working on metric spaces, where similarities are measured using a distance function. We will explain also how to use a metric index tree, *M*-tree [CIACCIA_97], to characterize a high-dimensional dataset, allowing estimation of selectivity for point and range queries. That is, we will be able to estimate the search performance on *M*-tree. In this paper we are focusing on range queries, but this approach can be used for other types of queries, as well as nearest neighbor, spatial joins, etc.

Interesting metric trees such as mvp-tree (multi-vantage point tree) [BOZKAYA_97], vp-tree (vantage-point [CHIUEH_94] and GNAT [BRIN_95] have been proposed lately.The mvp-tree and the vp-tree are static and balanced structures. Both use the concept of vantage points and relative distances to partition the data space. The mvp-tree chooses the vantage points and calculates the distances only once, when it is built. This approach minimizes the cost for distance calculations when a range query is asked. It was shown that for high-dimensional Euclidian vectors, the mvp-tree outperforms the vp-tree, and obtains around a 20-30% gain in efficiency [Bozkaya_97].

The latest representative of metric trees is the *M*-tree. It is a balanced and dynamic metric tree, which is well suited to work with high-dimensional or non-dimensional data. An *M*-tree partitions the data set based on the relative distance of the objects. The objects are stored into fixed-size nodes, which correspond to constrained regions of the metric space. A complete description of the *M*-tree is given in [CIACCIA_97]. The authors also provide an analysis of M-trees in [CIACCIA_98]. However, there is no estimation of the probability distribution function of the distances there, which is our major contribution, as we describe in next sections.

The $\mathcal{M}$-tree is built by dividing the data into regions specified by hyper-spheres[1]. The center of each hyper-sphere is an object chosen from the data and it is called a *routing object* (see below). Such regions can also be overlapped or not. An $\mathcal{M}$-tree can store up to C entries per node. Thus, C is the *capacity* of the nodes. A leaf node has the following format:

$entry(O_j) = [\ O_j,\ oid\ (O_j),\ d(O_j,\ P(O_j)\ ]$

where: $oid\ (O_j)$ is the identifier of the object into the database, O_j is the object - the feature values of the object - and $d(O_j,\ P(O_j)$ is the distance between O_j and its parent $P(O_j)$.

An internal node of the M-tree stores a *routing object* O_r and a *covering radius* $r(O_r) > 0$. The O_r entry includes a pointer, $ptr(T(O_r))$, that indicates the root of the sub-tree $T(O_r)$ - the *covering tree* of O_r. The distance from the parent object $d(O_r,\ P(O_r)$ is also stored.The internal node of the M-tree has the format:

$entry(O_r) = [\ O_r,\ ptr\ (T(O_r)),\ r(O_r),\ d(O_r,\ P(O_r)\ ]$

It must be noted that, for all objects O_j in the covering sub-tree of O_r, the distance to the routing object O_r is less than the covering radius, i. e. $d(O_j,\ O_r) \leq r(O_r)$. This premise says that it is possible to narrow the data set to be searched, using the covering radius as a measurement of the proximity between the data under analysis.

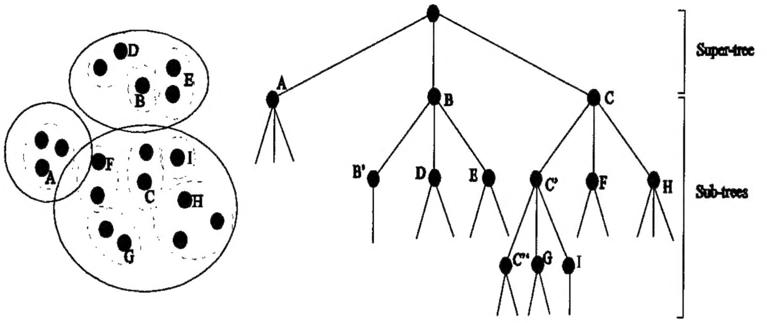

Figure 1a - First partitioning of the nodes to build the M-tree (it is not balanced).

For example, figure 1(a-b) shows a 2-D construction of a M-tree with fanout = 3 using the Euclidian distance. Figure 1a shows the first step of the algorithm that builds a $\mathcal{M}$-tree as created by Ciaccia and Patella, before it splits the nodes. At this first step, the object A, B, and C are selected by the algorithm as samples for the sub-trees. To simplify the drawing, it is assumed that a sample at a higher level is also a sample for lower levels, i. e. C=C'= C". The figure 1b shows the real $\mathcal{M}$-tree built for those elements. For more details about the algorithm to build a $\mathcal{M}$-tree see [Ciaccia_98].

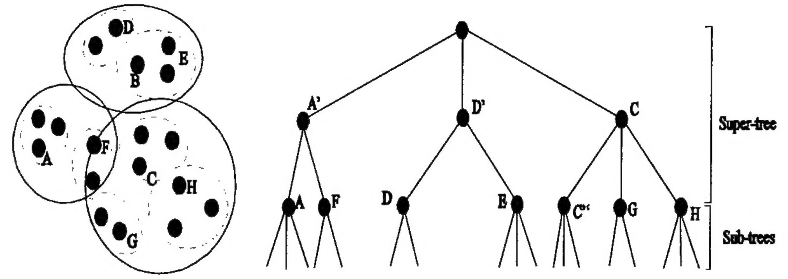

Figure 1b - M-tree built (it is now balanced).

In Table 1 we summarize all meaningful symbols used in this paper.

[1] The actual "shape" of the regions depends on the specific metric space. For example, using: $M=(\mathbb{R}^2, L_1)$ the regions are "diamonds", $M=(\mathbb{R}^3, L_2)$ the regions are "spheres", and $M=(\mathbb{R}^2, L_\infty)$ the regions are "squares".To simplify the comprehension of the ideas in this paper, we are using the Euclidian distance, and the shape of the regions of the M-tree as hyper-spheres.

Symbols	Definitions
$d(x,y)$	metric distance between objects x and y
D	Euclidean dimension
$\mathscr{D}$	*Distance exponent*
q_r	Range query for covering radius r
C_{eff}	Effective capacity of a metric-tree node (average number of points stored in a non-root node)
n_i	i^th node of the metric-tree
r_i	Covering radius of the n_i node of the metric-tree
r_q	Covering radius of a range query q_r
r_l	Average covering radius of a leaf node
r_h	Average covering radius of the nodes in the h^th level of the tree
H	Height of the metric-tree
No	Total number of objects in the metric-tree
No_h	number of objects stored in the h^th level of the metric-tree
N_l	Number of leaf nodes in the metric-tree
$N_{\mathcal{M}}$	Total number of nodes in the metric-tree
$DA_i(r)$	Number of disk access in the node n_i for a range query of radius r
$DA_{\mathcal{M}}(r)$	Number of disk access for all nodes in the metric-tree for a range query of radius r
$DA_{\mathcal{M}opt}(r)$	Number of disk access for all nodes in an optimal metric-tree for a range query of radius r

Table 1- Definition of Symbols

3 - Fundamental Observation: the "Distance Law"

In this section we explain our proposal, defining the concept of the *distance exponent*, and showing that it holds for a great variety of real data domains. The problem of interest is the distribution of distances between *No* given objects in a metric space for a specific distance *r*. Is it Gaussian? Is it Poisson? It turns out that, for several real and synthetic datasets (see Section 3.2) it follows a power law.

3.1 - Main Point

Our goal is to find a formula to estimate the number of neighbors of objects within a given distance *r* in a set of *No* objects. To make the discussion clearer, we introduce two definitions:

Definition 1 - The "distance plot" of a metric set is the number os pairs within distance *r* versus the distance *r*, where both axis are presented in log scale.

Notice that the distance plot is the "Accumulated Distribution Function" of the probability of a pair of objects to be within a distance *r*, that is $\Phi(r) = \int_{-\infty}^{r} P(\mathrm{d} \le r)\mathrm{d}r,\ \mathrm{d} = d(x,y),\ x \ne y$. Moreover, the number of neighbors of an object within a given distance *r* is $\overline{nb} = No \cdot \Phi(r)$.

Definition 2 - If a distance plot is linear for a range of scales, the slope of the line that best fits the distance plot is the *distance exponent* - $\mathscr{D}$.

Using these two definitions, we define the following "*Distance Law*", which gives an important stepping stone towards our goal.

Distance Law - *Given a set of No objects in a metric space with distance function d(x,y), the average number of distances less than a radius r follows a power law, i.e., the average number of neighbors* $\overline{nb}(r)$ *within a given distance r is proportional to r raised to* $\mathscr{D}$:

$$\boxed{No \cdot \Phi(r) = \overline{nb}(r) \propto r^{\mathscr{D}}}$$

If a dataset presents a metric way to evaluate the distance between any pairs of its objects, then this plot can always be drawn, even if it does not have a spatial property. Moreover, we are going to show that this distance plot presents an almost linear behavior for a large number of both real as well as synthetic datasets.

3.2 - Experimental Evidence

To evaluate the correctness of our proposal, we used a variety of data sets, both from the real world as well as synthetic ones. They are described follows:

	Data Set	No (# Objects)	Dimension	Distance Function	*Distance Exponent* - $\mathscr{D}$
Real Metric datasets	English	25143	NA (22*)	L_{Edit}	4.753
	Divina Commedia	12701	NA (15*)	L_{Edit}	4.827
	Decamerone	18719	NA (19*)	L_{Edit}	5.124
	Portuguese	21473	NA (26*)	L_{Edit}	6.686
	FaceIt	516	NA	Not divulged	5.301
Real spatial datasets	MGCount	15559	2	L_2	1.752
	Eigenfaces	11900	16	L_2	5.267
Synthetic datasets	Sierpinsky	9841	2	L_2	1.584
	2D Line	20000	2	L_2	0.989

Table 2 - Data sets used in the experiments. *Corresponds to the length of the longest word.

- "MGCounty" - a set of geographical data describing the coordinates of the road intersections in the Montgomery County - Maryland. The distance function[2] used was L_2 (Euclidian) and the number of objects is 15559 points in a two-dimensional space.
- "FaceIt" - a dataset constructed by a distance matrix given by the FaceIt version 2.51, a commercial product from Visionics Corp [VISIONICS_98]. The way the distance function work is unknown. The set of 1056 faces that generate this distance matrix was given by the Informedia project [WACTLAR_96] at Carnegie Mellon University.

[2] The L_p distance is defined as $L_p(x,y) = \left(\sum_{j=1}^{Dim} |x[j] - y[j]|^p \right)^{\frac{1}{p}}$.

- "Eigenfaces" - a set of 11900 face vectors from the Informedia project too. Each face was processed with the eigenfaces method [TURK_91], resulting in 16-dimensional vectors.The distance function used was L_2 over these vectors.
- "Divina Comedia" and "Decamerone" - two sets of Latin words (used in [CIACCIA_99]), with 12701 and 18719 objects respectively. The distance function used was L_{edit} [3].
- "English" - a set of 25143 objects from the English language dictionary. The distance function used was L_{edit}.
- "Portuguese" - a set of 21473 objects from the Portuguese language dictionary, including accented words. The distance function used was L_{edit}.
- "Sierpinsky" - a synthetic set of 9841 2-dimensional objects from the Sierpinsky triangle. The distance function used was L_2.
- "Line2D" - a synthetic set of 20K 2-dimensional objects from the Sierpinsky triangle. The distance function used was L_2.

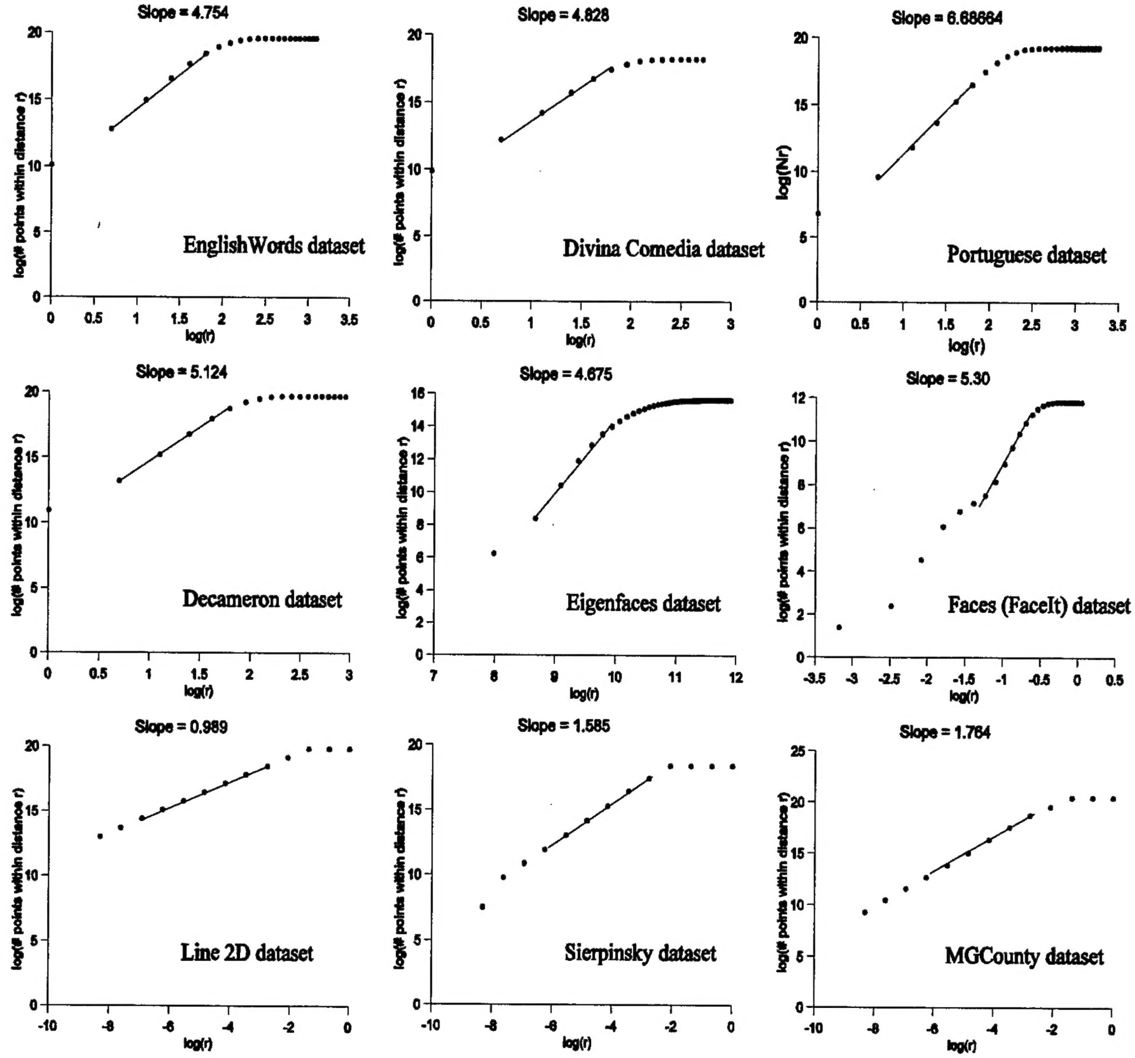

Figure 2 - Distance Exponent plots of diverse real and synthetic datasets, calculated using Log(Total of points within *r*) versus Log(*r*).

[3] L_{edit} or Levenshtein distance of two strings, L_{Edit} (x,y), is a metric which counts the minimal number of symbols that have to be inserted, deleted, or substituted, to transform *x* into *y* (e.g. L_{edit} ("head", "hobby") = 4 - three substitutions and one insertion).

Figure 2 shows the distance plots for all these datasets. As we stated in Definition 2 (previous Section), the slopes of these lines are the *distance exponents* - $\mathcal{D}$. From Figure 2, we observe the following:

- all our distance plots are linear for suitable ranges of *r* scales.
- it must be noted that the "FaceIt" dataset obeys our distance law, even though its distance function was not revealed to us.

Table 2 summarizes the relevant information used o calculate the *distance exponent* $\mathcal{D}$ from these datasets.

As different domains of data were used, different procedures to evaluate the distances in each dataset were used too. Clearly, the distance plot requires $O(No^2)$ distance computations, where *No* is the number of objects.

4 - Proposed Formula for Estimation of Disk Accesses for Range Queries

Our goal here is to obtain a quick way to estimate the number of disk accesses DA needed to answer a range query $Q_R(o_R, r_q)$ asking for the objects stored in a metric-tree within a given radius r_q from the query object o_R . Of foremost importance, we are devising a formula that can give this estimation from a set of global parameters that can be easily retrieved from an already built metric-tree[4]. Aiming for this goal, we present the following Lemma:

Lemma 1 (Disk accesses for Range Queries): For a range query $q_r = Q_R(o_R, r_q)$ with a covering radius r_q, the average number of disk accesses $DA_\sigma(r)$ in a given subset σ of the set $\mathcal{M}$ of nodes of a metric-tree is given by:

$$DA_\sigma(r) \approx \frac{1}{r_0^d} \sum_i^\sigma (r_i + r_q)^{\mathcal{D}} \tag{1}$$

Proof: The volume of a hyper-sphere of dimension *D* and radius *R* is $K_D * R^D$, where K_D is a constant (e.g., $D=2 \rightarrow V_2 = V(\text{Circle}) = \pi R^2$, $D=3 \rightarrow V_3 = V(\text{Sphere}) = \frac{3}{4}\pi R^3$, etc.). The expected probability $P_i(0)$ of any point query q_0 to fetch a given node n_i can be measured by taking the (hyper-)volume covered by that node divided by the overall volume covered by the metric-tree, that is:

$$P_i(0) = \frac{Volume(n_i)}{Volume(n_0)} = \frac{K_D r_i^D}{K_D r_0^D} = \frac{r_i^D}{r_0^D} \tag{2}$$

We assume that the dataset behaves as a manifold with intrinsic dimensionality $\mathcal{D} \in \mathbb{R}$. Thus:

$$P_i(0) \approx \frac{r_i^{\mathcal{D}}}{r_0^{\mathcal{D}}} \tag{3}$$

Therefore, Eq. 3 can be generalized to predict the average number of disk accesses needed to retrieve the node n_i by a point query $q_0 = Q_R(o_R, 0)$ in a metric-tree storing a set of points following any non-integer dimension $\mathcal{D}$.

$$DA_i(0) \approx \frac{r_i^{\mathcal{D}}}{r_0^{\mathcal{D}}} \tag{4}$$

[4] Inside this section, the term metric-tree and tree will be used interchangeably.

The expected number of disk accesses $DA_\sigma(0)$ in any subset σ of the set of tree nodes, to answer a point query q_0, is the sum of the probabilities to access each node in the set σ. This is given by:

$$DA_\sigma(0) \approx \sum_i^\sigma \frac{r_i^{\mathscr{D}}}{r_0^{\mathscr{D}}} = \frac{1}{r_0^{\mathscr{D}}} \sum_i^\sigma r_i^{\mathscr{D}} \qquad \textbf{(5)}$$

where r_i is the covering radius of each node into the subset σ.

A range query of radius r_q is equivalent to a point query over a modified metric-tree, where the covering radius of each node is enlarged by the query radius r_q. The summation of all nodes with enlarged radii (r_i+r_q) on Eq. 5, gives Eq. 1.

QED

Lemma 1 enables the calculation of the number of required disk accesses for a range query over the whole tree by making σ the set of all nodes of the tree. However, as it requires the covering radius r_i of each node of the tree, it is not a fast way to predict the average disk accesses needed in each query. So a further manipulation is needed.

4.1 - A fast way to predict the average number of disk accesses

In this section we are going to derive a formula to predict the number of disk accesses for range queries when these points are already stored in a metric-tree. Before that, the following definition is needed.

Definition 3: The *effective capacity* C_{eff} for the $\mathcal{M}$-tree nodes is the average number of objects stored in a non-root node of the tree. That is, given that each node can hold up to C objects, but has an average utilization of $u\%$, the effective capacity C_{eff} is:

$$C_{eff} = C * u$$

Considering that: a tree has H levels, the root is at level $h=0$ and the leaves are at level $h=H$-, we can state the following Lemma.

Lemma 2:. The number of objects that can be stored in each level of the tree is:

$$No_h = C_{eff}^{\,h+1} \quad , \quad h = 0, 1, \ldots H-1 \qquad \textbf{(6)}$$

Proof: Assuming a tree with fan-out C_{eff} at every node we have Eq. 6.

QED

In a metric-tree, each object is copied onto the leaf nodes, even when it is used as a split object in a non-leaf node, so that all objects appear in the leaf nodes. That is, considering $h=H-1$ for leaf nodes, $No_{H-1} = No_l = No$. Then, using Lemma 2 we have:

$$C_{eff} = No^{1/H} \qquad \textbf{(7)}$$

We assume that the algorithm to build the metric-tree is 'good', that is: that the number of objects in any node is approximately the same, that each node covers the closest objects in each region, and that there is a minimum overlap of the covering area between "sibling" nodes at the same level. Given that, we can state the following two Lemmas.

Lemma 3: The average covering radius r_h of a node at a level h is given by:

$$r_h = \sqrt[\mathcal{D}]{No^{\frac{-h}{H}}} \qquad \textbf{(8)}$$

Proof: We start this proof calculating the covering radius of the leaf nodes. Given the total number of objects *No* into the $\mathcal{M}$-tree, the approximate number of leaf nodes can be expressed as:

$$N_l = {No}/{C_{eff}} \qquad \textbf{(9)}$$

The typical assumption in the analysis of index trees [FALOUTSOS_94] is that the trees are "good", that is, the nodes correspond to bounding shapes (rectangles, spheres, etc) that are tight. In our case, the bounding shapes are spheres. Let r_l be the average radius of the nodes at the leaf level. A *distance exponent* $\mathcal{D}$ implies that our set of objects behaves like a manifold of dimensionality $\mathcal{D}$. In that case, the number N_l of spheres of radius r_l that are required to cover the *No* objects would be [SCHROEDER_91]:

$$N_l = {1}/{r_l^{\mathcal{D}}} \qquad \textbf{(10)}$$

that, combined with Eq. 9, gives the average covering radius of a leaf node as:

Assuming that the average fan-out of the tree is C_{eff} at every level (N_h=N_{Oh}/C_{eff}), the same process can be used to estimate the average covering radius r_h of the nodes in the hth level.

QED

$$r_l^{\mathcal{D}} = \frac{C_{eff}}{No} = \frac{No^{\frac{1}{H}}}{No} \quad \Rightarrow \quad r_l = \sqrt[\mathcal{D}]{No^{\frac{1-H}{H}}} \qquad \textbf{(11)}$$

Lemma 4: For an optimal metric tree, the total number of disk accesses $DA_{\mathcal{M}\text{-}opt}(r)$ on all nodes of the metric-tree which are needed to answer any range queries q_r with covering radius r can be estimated as:

$$\boxed{DA_{\mathrm{Mopt}}(r) \approx \frac{1}{r_0^{\mathcal{D}}} \cdot \sum_{h=0}^{H-1} No^{\frac{h}{H}} \left(\sqrt[\mathcal{D}]{No^{\frac{-h}{H}}} + r_q \right)^{\mathcal{D}}} \qquad \textbf{(12)}$$

Proof: Lemma 1 gives the predicted number of disk accesses for any sub-set of metric-tree nodes, like the full set of nodes or the leaf nodes. Given that, all nodes of each level have the same radius r_l,the summation in Lemma 1 turns into a count of the number of nodes in this level. Combining Lemma 1 with Eq. 9 and Eq. 11, it can estimate the average number of access in leaf nodes of the tree for a query q_r:

$$DA_{\mathrm{leave}}(r_q) \approx \frac{1}{r_0^{\mathcal{D}}} \sum_{i=1}^{N_l} (r_l + r_q)^{\mathcal{D}} = \frac{1}{r_0^{\mathcal{D}}} \left[\frac{No}{C_{eff}} \left(\sqrt[\mathcal{D}]{No^{\frac{1-H}{H}}} + r_q \right)^{\mathcal{D}} \right] \Rightarrow$$

$$DA_{\mathrm{leave}}(r_q) \approx \frac{No^{\frac{H-1}{H}}}{r_0^{\mathcal{D}}} \left(\sqrt[\mathcal{D}]{No^{\frac{1-H}{H}}} + r_q \right)^{\mathcal{D}} \qquad \textbf{(13)}$$

It must be noted that Eq. 13 holds for all levels of the tree but the root, as the root does not always hold the average of C_{eff} number of objects. However, calculating the C_{eff} from Eq. 7 spreads this difference to the other levels, so a good approximation of the average radii over all H levels can be achieved using all levels, including the root one. Thus, the total number of disk accesses $DA_{\mathcal{M}}(r)$ over the nodes in any level h of the tree needed to answer any range queries q_r with covering radius r can be estimated as:

$$DA_h(r) \approx \frac{1}{r_0^{\mathcal{D}}} \cdot No^{\frac{h}{H}} \cdot \left(\sqrt[\mathcal{D}]{No^{\frac{-h}{H}}} + r_q \right)^{\mathcal{D}} \tag{14}$$

Applying this equation over all levels of the tree gives Equation 12.

QED

Notice that, if we do not have an optimal metric tree: $DA_M(r) > DA_{Mopt}(r)$.

Lemma 4 represents the predicted number of disk accesses needed to answer a range query $Q_R(o_R, r)=q_r$ of a given covering radius r in a metric-tree, given that the *distance exponent* $\mathcal{D}$ is known. It uses only global parameters of that tree, as no information about each particular node is needed.

4.2 - Using a metric-tree to predict the *distance exponent* $\mathcal{D}$of the stored objects

Lemma 4 can be used to estimate the average number of disk accesses needed to retrieve the set of objects that answer a given range query $Q_R(o_R, r_q)$. However, the estimation depends on previous knowledge of the *distance exponent* of the whole set of objects already stored in the tree. As seen, this value can be obtained by calculating the distances between all pairs of objects. However, although generic, this is a very time-consuming process because it is an $O(No^2)$ algorithm.

We propose here an alternative way to *estimate* the *distance exponent* $\mathcal{D}$ of a set of objects stored in a metric-tree, using the tree itself. In fact, Eq. 8 holds for all levels of the metric-tree but the root node. That is:

$$\begin{aligned} r_h &= \sqrt[\mathcal{D}]{No^{\frac{-h}{H}}} \Rightarrow \\ \mathcal{D} &= -\frac{h}{H} \cdot \frac{\log No}{\log r_h}, \quad (h = 1, \dots, H-1) \end{aligned} \tag{15}$$

Using this equation, a set of approximate values for the *distance exponent* $\mathcal{D}$ can be retrieved from the tree, one for each non-root level of the tree. By averaging this set of values, an approximated value for $\mathcal{D}$, we are calling $\hat{\mathcal{D}}$, can be estimated from the tree itself. To obtain this value, the tree must be traversed, averaging the value r_h in each level of the tree. We have calculated the arithmetic and the geometric average, applied over both the average of the individual radii of each node in each level $\overline{r_h}$ measured in all levels of the tree, as well as the least squares regression line applied over the whole set of nodes of the tree. We found that the arithmetic average of the formula in each level, using the measured radii $\overline{r_h}$, give the best results. That is:

$$\boxed{\hat{\mathcal{D}} = \frac{\sum_{h=1}^{H-1} -\frac{h}{H} \cdot \frac{\log No}{\log \overline{r_h}}}{H-1}} \tag{16}$$

5 - Experimental Results

We carried out several experiments in order to compare our analytical results with the results given by an $\mathcal{M}$-tree. All experiments were run on 450MHz Pentium II machines. They were written in C++ language. In this section we describe the results obtained showing that they support our predictions within a good margin of error . Interesting observations are discussed following:

5.1 - The accuracy of *distance exponents* $\mathcal{D}$ and $\hat{\mathcal{D}}$.

We can compare the results obtained from the calculation of the *distance exponent* $\mathcal{D}$ and the *estimated distance exponent* $\hat{\mathcal{D}}$ from the $\mathcal{M}$-tree. Table 3 summarizes these results, and we can see that the error margin is below 20% for all datasets, and with two exceptions, below 10%.

Data Set	Size (# Objects) *No*	Number of Levels *H* ($\mathcal{M}$-tree)	*calculated distance exponent* $\mathcal{D}$	*estimated distance exponent* (measured from $\mathcal{M}$-tree) $\hat{\mathcal{D}}$	Error %
English	25143	4	4.753	3.979	16
Divina Commedia	12701	4	4.827	5.134	6
Decamerone	18719	4	5.123	5.647	10
Portuguese	21473	5	6.686	6.371	5
MGCount	15559	3	1.752	1.452	17
Eigenfaces	11900	4	5.267	4.770	9
FaceIt	1056	3	6.821	6.471	5
Sierpinsky	9841	3	1.584	1.525	4

Table 3 - *Distance exponents* calculated and estimated from the datasets used in the experiments.

5.2 - The computing time to obtain *distance exponents* $\mathcal{D}$ and $\hat{\mathcal{D}}$.

Table 4 shows the computing time measured to calculate the *distance exponent* $\mathcal{D}$ and the *estimated distance exponent* $\hat{\mathcal{D}}$. It should be noted that if a given application already has the $\mathcal{M}$-tree built, time needs only to be spent obtaining $\hat{\mathcal{D}}$ .The time to obtain $\hat{\mathcal{D}}$ from the $\mathcal{M}$-tree is too small (from fractions of seconds up to few seconds for word datasets) comparing with the computing time to calculate $\mathcal{D}$ numerically.

Data Set	Time to build the $\mathcal{M}$-tree (sec.)	Time to obtain $\hat{\mathcal{D}}$ from $\mathcal{M}$-tree (sec.)	Time to calculate $\mathcal{D}$(sec.)
English	171.31	1.34	7939.21
Decamerone	130.73	1.89	2974.16
Divina Commedia	97.94	2.97	1085.55
Portuguese	152.50	1.16	8641.57
MGCount	30.91	0.55	575.98
Eigenfaces	77.13	0.59	23152.75
FaceIt	12.73	0.04	10653.79
Sierpinsky	14.53	0.33	84.51
2D Line	51.87	0.69	305.51

Table 4- Time (in seconds) needed to obtain *distance exponents* $\mathcal{D}$ and $\hat{\mathcal{D}}$.

5.3 - The accuracy of our proposed selectivity formulas.

Here we compare the results obtained from the measurement of real queries in a $\mathcal{M}$-tree with the estimation made using Lemma 4. For this measurement, we plot the average disk accesses evaluated from the $\mathcal{M}$-tree to answer 500 queries with each given radius. Figure 3 shows the measurements obtained from the various datasets and the respective estimations using both the calculated *distance exponent* $\mathcal{D}$ and the *estimated distance exponent* $\hat{\mathcal{D}}$, versus the query radius r_q. The graphs from Figure 3 are shown in log-log scales. The

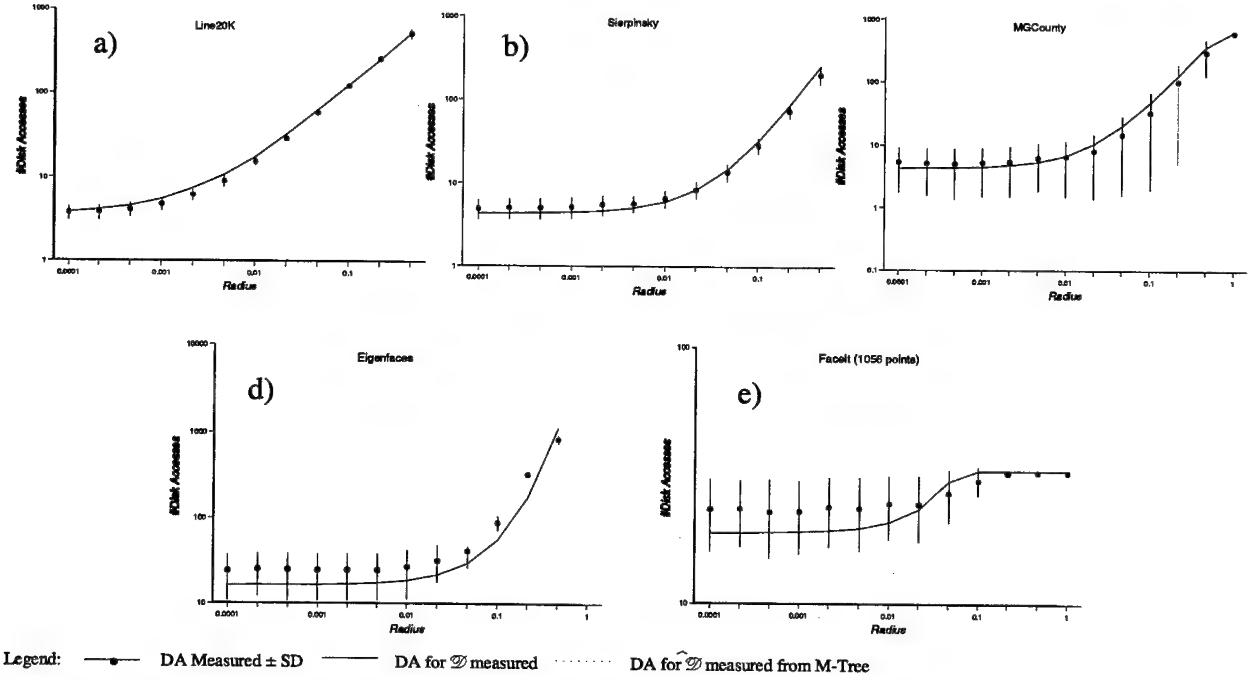

Figure 3 - Plots in log-log scale comparing actual number of disk accesses (marked by big dots) inside the standard deviation (error bar), with the estimated disk accesses using $\mathcal{D}$ (dotted line) and estimated disk accesses using $\hat{\mathcal{D}}$ (continuous line).

solid line represents the average number of disk accesses obtained using the *estimated distance exponent* $\hat{\mathscr{D}}$ and the dotted line represents the average number of disk accesses obtained using *distance exponent* $\mathscr{D}$. The measured disk accesses obtained from the $\mathcal{M}$-tree are presented by the big markers within a bar that represents the standard-deviation for these data. As we can see in Figure 3, the results for the disk access measurements for "Sierpinsky", "Line2D", "MGCounty", "EigenFaces" and "FaceIt" datasets follow the prediction of the number of disk accesses stated by the proposed formula (Eq. 12). We can see also that the prediction is within an error margin of one standard deviation displacement from the points.This is a great result, showing also that the $\mathcal{M}$-tree constructed for these datasets are nearly optimal.

Lemma 4 holds, that is, gives a good approximation, if the metric tree is optimal. Unfortunately, this not happened with our word datasets. We conjecture that this is due to the large number of pairs of words within the same distance. This leads to a large overlapping of covering radius within the same level, so the number of disk accesses needed to answer a given query becomes quite large. In fact, the measurements shown that with the words datasets, the $\mathcal{M}$-tree behaves nearly as a sequential scanning. For example, to answer a query asking for English words with distance equal to 4 (words must differ by a total of 4 operations - either insertions, deletions or substitutions of characters), it needs to read an average of 93% of the nodes. We tried to test this conjecture, changing the distance function L_{edit} by giving weights to the substitute, insert and delete operations, so that the distance function returns a broader response domain. Fig. 4 compares the result for the "English" dataset. It has two pairs of curves, one for the original distance function L_{edit}, and the other for the modified one. The dashed-dotted line represents the number of disk accesses measured using the original L_{edit} distance function, whereas the error bars/dots convention represents the number of disk accesses measured using the modified L_{edit}. The estimated number of disk accesses are shown using only the predicted $\mathscr{D}$, both for the original L_{edit} distance function in dashed line, and for the modified L_{edit} distance function in solid line. It can be seen that, when a distance function gives a broader domain of distance values, the measurements are better due to a lesser number of disk accesses, and the predicted number of disk access tends to get closer the real measurements. The radius axis in this figure is in linear scale, because the queries for words are made asking for words with 1, 2, 3... characters in difference, that is, in a linear way. The other word-datasets have similar behavior, so they are not shown here. As we can also see, the modified distance function for words leads to a better tree.

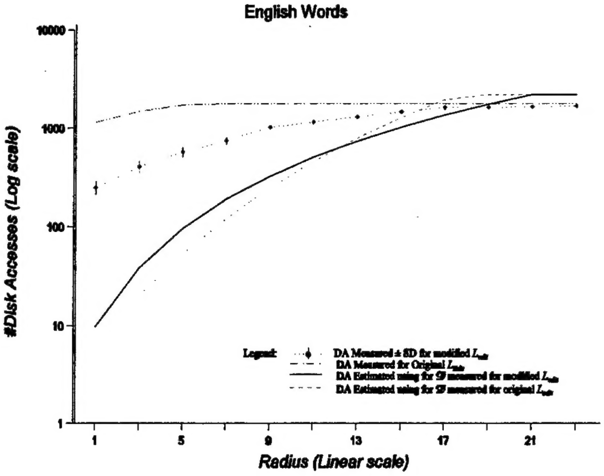

Figure 4 - Plot in linear-log scale comparing actual and estimated number of disk accesses. Actual disk accesses are shown for: modified L_{Edit} Distance function(marked by big dots) inside the standard deviation (error bar); and original L_{Edit} Distance. Estimated disk accesses are show using the predicted $\mathscr{D}$ for: modified L_{Edit} Distance (continuous line); and original L_{Edit} Distance. (dotted line)

From the previous observation, we can notice that the detected problem is due to the distance function used, which does not yield the construction of an optimal tree. This indeed shows that our approach becomes an effective way to evaluate whether the $\mathcal{M}$-tree is a good data structure for implementing an index structure for a specific dataset domain with a given distance function. Moreover, the presented formulas can also spot bad trees: if a tree gives a performance that is worse than our predicted disk accesses, we can suspect that the tree is sub-optimal. This occurs with the word datasets.

6 - Conclusions

This paper focuses on selectivity estimations for metric datasets. To the best of our knowledge, this is the first and only attempt on modeling the probability density function (PDF) of the distances. Our major contribution is the 'Distance Law', an empirical power law that holds for the PDF of surprisingly many real datasets. The exponent of our power law, the 'distance exponent', is the key to solve the selectivity estimation problems that motivated this work. Additional contributions are:

- the derivation of formulas that estimate the number of disk accesses for an optimal metric tree. These formulas use the distance exponent $\mathscr{D}$, and they hold for metric and vector datasets alike. Our formulas are useful for query optimization, as well as for identifying sub-optimally constructed index trees.
- a fast algorithm to compute the *distance exponent*, from a given (metric) index tree. Our algorithm gives accurate estimates, for a fraction of the time that a naive algorithm would need.
- experiments on real datasets, illustrating the accuracy of our 'Law', as well as the accuracy of our disk access formulas. Excluding degenerate, sub-optimal trees, our estimation formulas are always within one standard deviation from the measurements.

Future work could further explore the use of our 'Distance Law' and the distance exponent, to analyze nearest neighbor queries [ROUSSOPOULOS_95], all-pairs queries and so on, on metric datasets.

Acknowledgments

We are grateful to Pavel Zezula, Paolo Ciaccia for giving us the code of the $\mathscr{M}$-tree and to Marco Patella, for helping us with the code of the $\mathscr{M}$-tree and for providing the datasets from [CIACCIA_99]. Also to Ricky Houghton and Ellen Hughes for providing the face-distance data from the Informedia project at Carnegie Mellon university. Thanks to Maria das Graças Nunes and Ricardo Hasegawa for giving us a subset of the Portuguese dictionary used in ReGra project [MARTINS_98] at University of São Paulo - Brasil.

References

[BECKMANN_90] N. Beckmann, H.P. Kriegel,; R. Schneider, B. Seeger -*"The R*-tree: An Efficient and Robust Access Method for Points and Rectangles"*, Proc. ACM SIGMOD Conference, pp. 322-331, Atlantic City, May 1990.

[BERCHTOLD_97] S. Berchtold, D.A. Keim, H. P. Kriegel - *"A Cost Model for Nearest Neighbor search in high-dimensional data space"*, Proc. ACM PODS'97, pp. 78-86, Tucson, Arizona, 1997.

[BOZKAYA_97] T. Bozkaya, M. Ozsoyoglu - *"Distance-Based Indexing for High-Dimensional Metric Spaces"*, Proc. ACM SIGMOD Conference, pp. 357-368, Tucson, Arizona, 1997.

[BRIN_95] S. Brin- *"Near neighbor search in large metric spaces"*, Proc. VLDB International Conference, pp.574-584, Zurich, Switzerland, Sept. 11-15 1995.

[CHIUEH_94] T. Chiueh - *"Content-Based Image Indexing"*, Proc. VLDB International Conference, pp. 582-593, Santiago, Chile, September 1994.

[CIACCIA_97] P. Ciaccia, M. Patella, P. Zezula - *"M-tree: An efficient access method for similarity search in metric spaces"*, Proc. VLDB International Conference, pp. 426-435, Athens, Greece, September 1997.

[CIACCIA_98] P.Ciaccia, M. Patela and P. Zezula - *"A Cost Model for Similarity Queries in Metric Spaces"*, Proc. ACM PODS, pp. 59-68, Seattle, Washington, 1998.

[CIACCIA_99] P. Ciaccia, A. Nanni, M. Patella - *"A Query-sensitive Cost Model for Similarity Queries with M-tree"*, Proc. 10th Australasian Database Conference (ADC'99), pp. 65-76, Auckland, New Zealand, January 1999.

[EVANGELIDIS_95] G. Evangelidis, D. Lomet and B. Salzberg - *"The hBP-tree: A Modified hB-trSupporting Concurrency, Recovery and Node Consolidation,* Proc. VLDB International Conference, pp. 551-561, Zurich, Switzerland, Sept. 11-15, 1995.

[FALOUTSOS_94] C. Faloutsos, L. Kamel - *"Beyond Uniformity and Independence: Analysis of R-tree Using the Concept of Fractal Dimension"*, Proc. ACM PODS, pp. 4-13, May 1994.

[GAEDE_98] V. Gaede, O. Gunther - *"Multidimensional Access Methods"*, ACM Computing Surveys, Vol. 30, No. 2, pp.170-231, June 1998.

[GONZALEZ_92] R. C. Gonzalez - *"Digital Imaging Processing"*, Addison-Wesley Publish. Co, 1992.

[GUTTMAN_84] A. Guttman - *"R-Tree: Adynamic Index Structure for Spatial Searching"*, Proc. of the 1984 ACM SIGMOD Conference, pp. 47-57, Boston, June 1984.

[HELLERSTEIN_95] Joseph M. Hellerstein, J. F. Naughton, A. Pfeffer: - *Generalized Search Trees for Database Systems*. Proc. VLDB International Conference, pp. 562-573, 1995.

[HUANG_97] Y.-W. Huang, N. Jing and E. A. Rundensteiner: "Spatial Joins Using R-trees: Breadth-First Traversal with Global Optimizations", Proc. VLDB International Conference pp. 396-405, Athens, Greece, 1997.

[KAMEL_93] I. Kamel and C. Faloutsos - *"On Packing R-trees"*, Proc. Second Int. Conference on Information and Knowledge Management (CIKM), pp. 490-499, Washington, DC, 1993.

[MARTINS_98] R.T. Martins, R. Hasegawa, M.G.V. Nunes, M.G.V, G. Montilha, and O.N Oliveira Jr. - *"Linguistic issues in the development of ReGra: a Grammar Checker for Brazilian Portuguese"*. Natural Language Engineering. Vol.4(4), December 1998.

[PAPADIAS_95] Dimitris Papadias, Yannis Theodoridis, Timos K. Sellis, Max J. Egenhofer - *"Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-trees"*, Proc. ACM SIGMOD Conference, pp 92-103, San Jose-CA, May, 1995.

[ROUSSOPOULOS_95] N. Roussopoulos and S. Kelley and F. Vincent - *"Nearest Neighbor Queries"*, Proc. ACM-SIGMOD Conference, pp. 71-79, San Jose, CA, May, 1995.

[SCHROEDER_91] M. Schroeder - *"Fractals, Chaos, Power Laws"*, W.H. Freeman and Company, 1991, pp. 41.

[SELLIS-87] T. Sellis, N. Roussopoulos, C. Faloutsos - *"The R+-tree: A Dynamic Index for Multi-dimensional Objects"*, Proc. VLDB Conference, pp. 507-518, Sept. 1987.

[TURK_91] M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, Vol. 3(1), pp. 71-86, 1991.

[UDUPA_91] J.K.Udupa, G.T. Herman - *"3D Medical Imaging"*, CRC Press, 1991.

[VISIONICS_98] Visionics Corp. - Available at http://www.visionics.com/live/frameset.html (12-Feb-1999).

[WACTLAR_96] H. D. Wactlar, T. Kanade, M. A. Smith and S. M. Stevens - *"Intelligent Access to Digital Video: Informedia Project"* , IEEE Computer, vol. 29 (3), pp. 46-52, May 1996.

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890